



Control Structures

<https://www.r-bloggers.com/control-structures-loops-in-r/>

License

These materials are available under the Creative Commons Attribution NonCommercial ShareAlike (CC-NC-SA) license
<https://creativecommons.org/licenses/by-nc-sa/3.0/>

Control Structures

Control structures in R allow you to control the flow of execution of the program, depending on runtime conditions. Common structures are

- **if, else:** testing a condition
- **for:** execute a loop a fixed number of times
- **while:** execute a loop while a condition is true
- **repeat:** execute an infinite loop
- **break:** break the execution of a loop
- **next:** skip an iteration of a loop
- **return:** exit a function

Most control structures are not used in interactive sessions, but rather when writing functions or longer expressions.

Control Structures: if

```
if(<condition>) {  
    ## do something  
} else {  
    ## do something else  
}  
  
if(<condition1>) {  
    ## do something  
} else if(<condition2>) {  
    ## do something different  
} else {  
    ## do something different  
}
```

if

This is a valid **if/else** structure.

```
if(x > 3) {  
  y <- 10  
} else {  
  y <- 0  
}
```

So is this one.

```
y <- if(x > 3) {  
  10  
} else {  
  0  
}
```

if

Of course, the **else** clause is not necessary.

```
if(<condition1>) {  
}  
if(<condition2>) {  
}
```

for

for loops take an iterator variable and assign it successive values from a sequence or vector. **For** loops are most commonly used for iterating over the elements of an object (list, vector, etc.)

```
for(i in 1:10) {  
    print(i)  
}
```

This loop takes the *i* variable and in each iteration of the loop gives it values 1, 2, 3, ..., 10, and then exits.

for

These three loops have the same behavior.

```
x <- c("a", "b", "c", "d")
```

```
for(i in 1:4) {  
  print(x[i])  
}
```

```
for(i in seq_along(x)) {  
  print(x[i])  
}
```

```
for(letter in x) {  
  print(letter)  
}
```

```
for(i in 1:4) print(x[i])
```

Nested for loops

for loops can be nested.

```
x <- matrix(1:6, 2, 3)
```

```
for(i in seq_len(nrow(x))) {  
  for(j in seq_len(ncol(x))) {  
    print(x[i, j])  
  }  
}
```

Be careful with nesting though. Nesting beyond 2–3 levels is often very difficult to read/understand.

While

While loops begin by testing a condition. If it is true, then they execute the loop body. Once the loop body is executed, the condition is tested again, and so forth.

```
count <- 0
while(count < 10) {
  print(count)
  count <- count + 1
}
```

While loops can potentially result in infinite loops if not written properly. Use with care!

While

Sometimes there will be more than one condition in the test.

```
z <- 5
while(z >= 3 && z <= 10) {
  print(z)
  coin <- rbinom(1, 1, 0.5)
  if(coin == 1) { ## random walk
    z <- z + 1
  } else {
    z <- z - 1
  }
}
```

Conditions are always evaluated from left to right.

Repeat

Repeat initiates an infinite loop; these are not commonly used in statistical applications but they do have their uses. The only way to exit a **repeat** loop is to call `break`.

```
x0 <- 1
tol <- 1e-8

repeat {
  x1 <- computeEstimate()

  if(abs(x1 - x0) < tol) {
    break
  } else {
    x0 <- x1
  }
}
```

Repeat

The loop in the previous slide is a bit dangerous because there's no guarantee it will stop. Better to set a hard limit on the number of iterations (e.g. using a for loop) and then report whether convergence was achieved or not.

next, return

next is used to skip an iteration of a loop

```
for(i in 1:100) {  
  if (i <= 20) {  
    ## Skip the first 20 iterations  
    next  
  }  
  ## Do something here  
}
```

return signals that a function should exit and return a given value

Control Structures

Summary

- Control structures like **if**, **while**, and **for** allow you to control the flow of an R program
- Infinite loops should generally be avoided, even if they are theoretically correct.
- Control structures mentioned here are primarily useful for writing programs; for command-line interactive work, the ***apply** functions are more useful.